UNIVERSITY OF CAMBRIDGE

Active Memory Networks for Language Modelling

O. Chen, A. Ragni, X. Chen, M.J.F. Gales

Cambridge University Engineering Department (CUED)

Structure of This Talk

- Statistical Language Modelling (2 min)
- Background (10mins)
  - recurrent neural networks
  - memory networks
- Active Memory Networks (15mins)
  - model architecture
  - training and regularization
- Experimental Results (10mins)
- Conclusions and Future Work (5mins)
- Questions (15mins)

Statistical Language Modelling

Statistical Language Modelling

- A language model (LM) models the log-probability of a word sequence

$$\log P(\mathbf{w}) = \sum_{t=1}^{T} \log P(w_t | w_{t-1}, \ldots, w_1)$$

- Prob. of the next word is conditioned on the history of all words seen by model.
- Quick example:

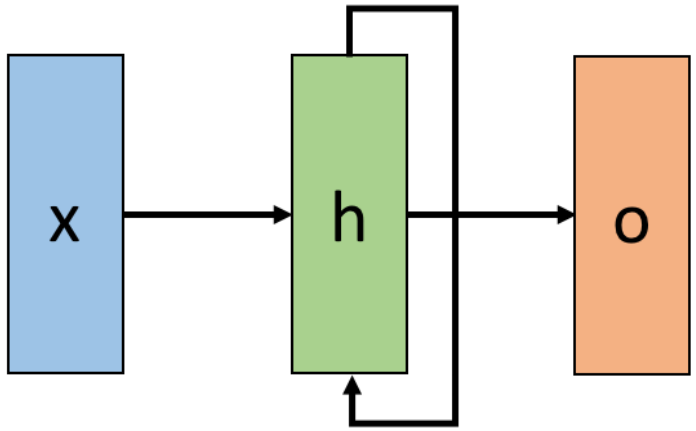
> the markets were open for most of the morning but closed in the afternoon due to the [BLANK]

Why Language Models?

- Rescore a set of sentence hypotheses for automatic speech recognition (ASR) or machine translation.
  - State-of-the-art ASR pipelines typically uses a LM.
- Test new ML models due to simplicity/abundance of text data and simple evaluation metric.

Background

Recurrent Neural Network (RNN)



RNN (Elman-variant)

- Popular for modelling sequential data.
- x is the input vector, typically an one-hot vector for the input word.
- h is the hidden state, which depends on both the input and the previous hidden state.

$$h_{t+1} = f(W_x x_t + W_h h_t + b)$$

- o is the output vector.

RNN for Language Modelling

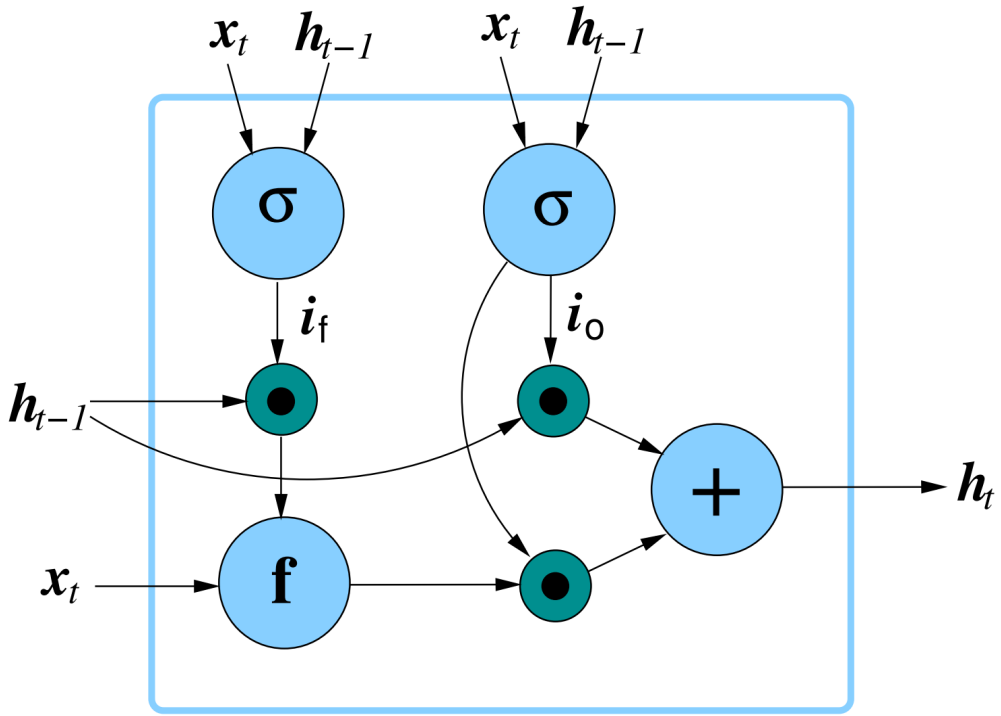
- Uses the (complete) word history.
  - approximated using a continuous vector representation given by hidden state.
  - output word probabilities are computed using a softmax activation.

$$
\begin{aligned}
P(\mathbf{w}) &= \prod_{t=1}^{T} P(w_t | w_{t-1}, ..., w_1) \\
&\approx \prod_{t=1}^{T} P(w_t | h_{t-1}) \\
&\approx \prod_{t=1}^{T} \mathrm{Softmax}(W_h h_{t-1} + b_h)
\end{aligned}
$$

Issues with RNN

- Difficulties remembering information.
  - Addressed by using network gates.
- Does not adapt to the topics in the data.
  - standard approach is to append the topics features from LDA to the input vector.
- Hard to interpret what was learned in the hidden state.

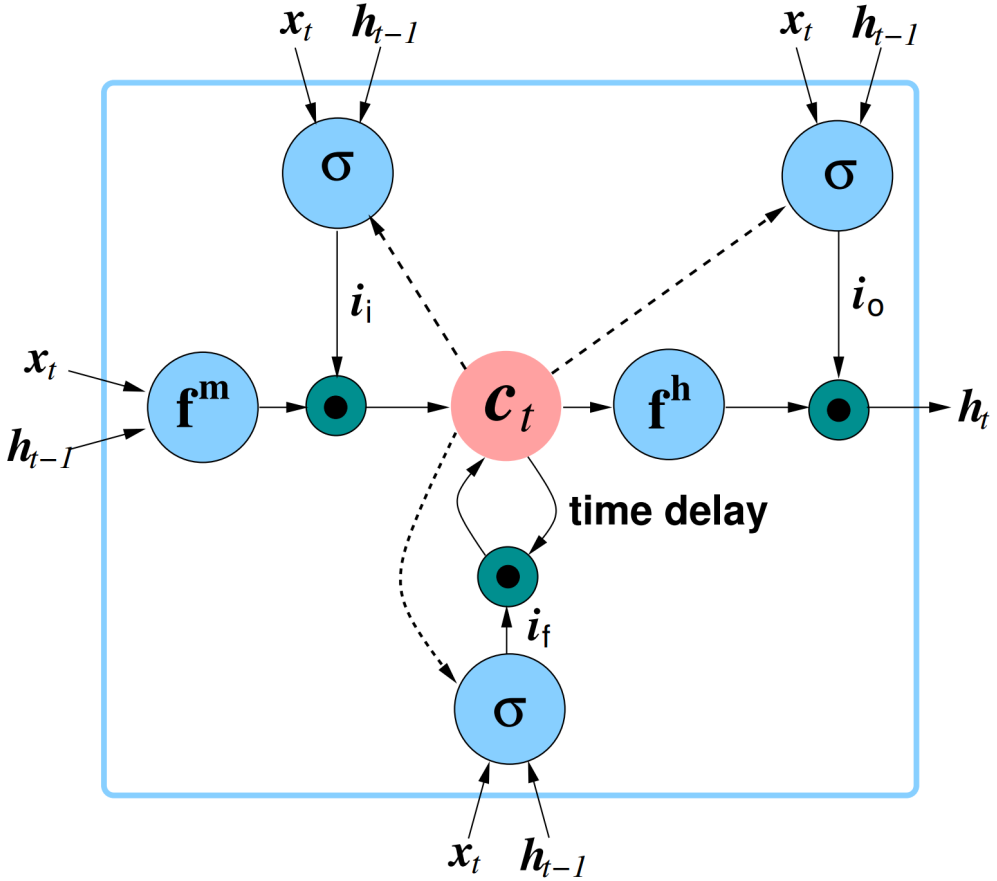
RNN-variants: Gated Recurrent Unit (GRU)



- Commonly used to improve memory retention.
- Consists of a forget gate and output gate.

$$
\begin{aligned}
\boldsymbol{i}_{\mathrm{f}} &= \sigma\left(\mathbf{W}_{\mathrm{f}}^{\mathrm{f}}\boldsymbol{x}_t + \mathbf{W}_{\mathrm{f}}^{\mathrm{r}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathrm{f}}\right) \\
\boldsymbol{i}_{\mathrm{o}} &= \sigma\left(\mathbf{W}_{\mathrm{o}}^{\mathrm{f}}\boldsymbol{x}_t + \mathbf{W}_{\mathrm{o}}^{\mathrm{r}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathrm{o}}\right) \\
\boldsymbol{y}_t &= \mathrm{f}\left(\mathbf{W}_{\mathrm{y}}^{\mathrm{f}}\boldsymbol{x}_t + \mathbf{W}_{\mathrm{y}}^{\mathrm{r}}\left(\boldsymbol{i}_{\mathrm{f}} \odot \boldsymbol{h}_{t-1}\right) + \boldsymbol{b}_{\mathrm{y}}\right) \\
\boldsymbol{h}_t &= \boldsymbol{i}_{\mathrm{o}} \odot \boldsymbol{h}_{t-1} + \left(1 - \boldsymbol{i}_{\mathrm{o}}\right) \odot \boldsymbol{y}_t
\end{aligned}
$$

RNN-variants: Long Short Term Memory Networks (LSTM)



- Similar to GRU; uses (more) gates to modulate information flow.

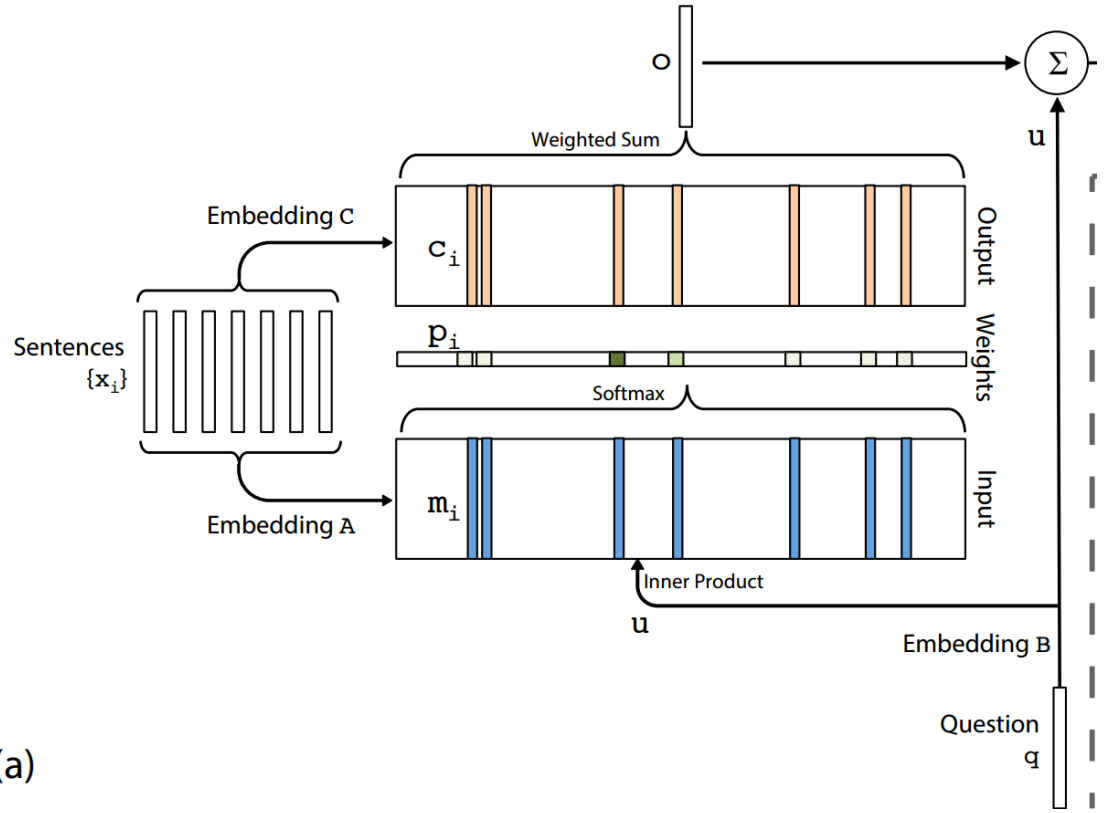
Forget gate (i_f), Input gate (i_i), Output gate (i_o)

$$
\begin{aligned}
i_f &= \sigma(W_f^f x_t + W_f^r h_{t-1} + W_f^m c_{t-1} + b_f) \\
i_i &= \sigma(W_i^f x_t + W_i^r h_{t-1} + W_i^m c_{t-1} + b_i) \\
i_o &= \sigma(W_o^f x_t + W_o^r h_{t-1} + W_o^m c_t + b_o)
\end{aligned}
$$

Memory Cell, history vector and gates are related by

$$
\begin{aligned}
c_t &= i_f \odot c_{t-1} + i_i \odot f^m(W_c^f x_t + W_c^r h_{t-1} + b_c) \\
h_t &= i_o \odot f^h(c_t)
\end{aligned}
$$

Memory Networks



(a)

- x_i are sentences (e.g. bag-of-words).
- c_i and m_i are vectors obtained from applying embedding matrices.
- Attention-mechanism:

$$v = [u^T m_1 \; .. \; u^T m_T]^T$$

$$p = \text{Softmax}(v)$$

- The final response vector:

$$o = \sum_i p_i c_i$$

Memory Networks for Language Modelling

- Uses a truncated word history approximation.

$$P(\mathbf{w}) = \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_1)$$
$$\approx \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n})$$

- Input is a sequence of words instead of sentences.
  - Memory cells hold word embeddings instead of sentence embeddings.
- Query vector u is fixed to some constant vector (e.g. 0.1).

Active Memory Networks (AMNs)

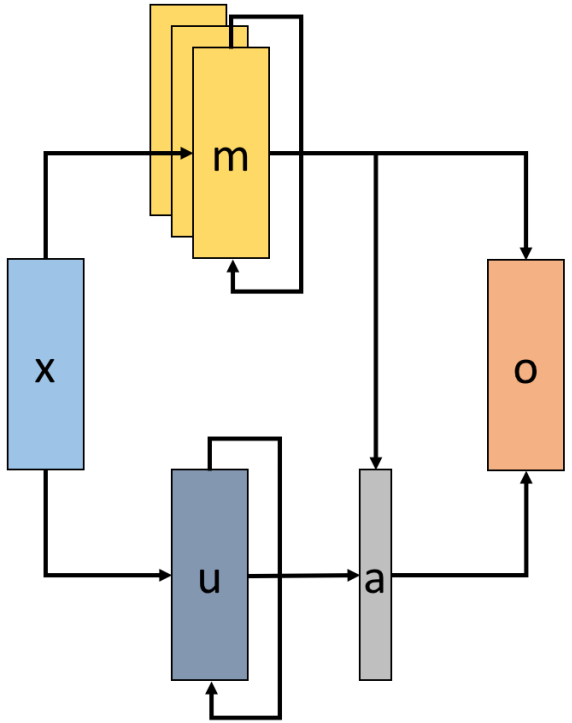
Motivation: Dealing with Context Ambiguity

- **Problem: topic/context of a sentence is not always clearly defined.**
- Consider the example earlier:

> the markets were open for most of the morning but closed in the afternoon due to the [BLANK]

- Interpretation 1: fruit stalls in open-air markets closing early due to a weather anomaly.
- Interpretation 2: stock markets closing early due to a sudden financial crash.
- Possible candidates for [BLANK]: “snow-storm” or “crash”.
- **Solution: keep track of multiple interpretations of word histories in memory.**

Active Memory Networks (AMNs)



AMN

- u_t is the controller vector; $m_t^{(i)}$ is the i[th] memory cell vector.

$$u_t = \mathrm{GRU}(u_{t-1}, x_t)$$

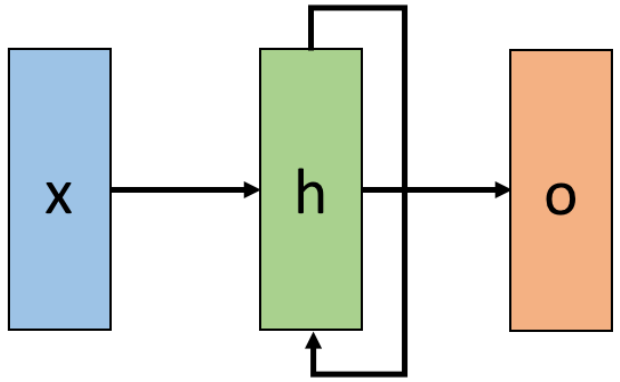
- $a_t^{(i)}$ is the attention value for the i[th] memory cell.

$$\beta_t^{(i)} = u_t \cdot m_t^{(i)}$$

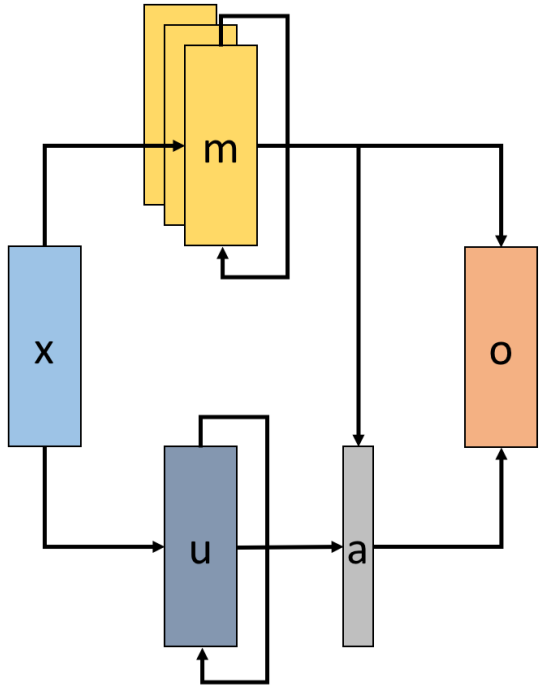
$$\alpha_t^{(i)} = \frac{\exp(\beta_t^{(i)})}{\sum_j \exp(\beta_t^{(j)})}$$

- $o_t = \sum_k a_t^{(i)} m_t^{(i)}$ is the response vector.

Comparison of AMN Architecture



RNN



AMN

AMN for Language Modelling

- LM approximation is same as RNN.

$$P(\mathbf{w}) = \prod_{t=1}^{T} P(w_t | w_{t-1}, ..., w_1)$$
$$\approx \prod_{t=1}^{T} \text{Softmax}(W_o o_{t-1} + b_o)$$

- Controller and memory cells each hold an unique, compact representation of the word history.
- Response vector o is used as input to a final softmax layer for word predictions.

Training the Attention-Mechanism

- Vanilla training with backpropagation = poor generalization performance.
  - model only trains small subset of memory cells.
- Consistent with the analysis of the derivative of the response vector w.r.t. weights for the memory cells.
  - weight-updates proportional to attention => only memory cells that get used are updated.

$$\frac{\partial o}{\partial w^{(k)}} = \alpha^{(k)} \frac{\partial m^{(k)}}{\partial w^{(k)}} \left(1 + \beta^{(k)} - \sum_{i=1}^{K} \alpha^{(i)} \beta^{(i)}\right)$$

Training the Attention-Mechanism: Annealing

- Solution 1: use annealing schedule for attention values.
  - Forces model to train with all memory cells in the initial training epochs.
  - T is high => attention is evenly distributed.
  - T is low => attention is focused.

$$\alpha_t^{(i)} = \frac{\exp(\beta_t^{(i)}/T_t)}{\sum_j \exp(\beta_t^{(j)}/T_t)}$$

$$T_{t+1} = \gamma \cdot T_t$$

Training the Attention-Mechanism: Dropout

- Solution 2: apply dropout to memory cells.

$$\hat{m}_t^{(i)} = W \begin{pmatrix} x_t \odot z_{m_t}^{(i)} \\ m_t^{(i)} \end{pmatrix}$$

- Regularizes both attention mechanism and memory vectors.

$$\hat{\beta}_t^{(i)} = u_t \cdot \hat{m}_t^{(i)}$$

$$\hat{\alpha}_t^{(i)} = \text{Softmax}(\hat{\beta}_t^{(i)})$$

$$\hat{o}_t = \sum_i \hat{\alpha}_t^{(i)} \hat{m}_t^{(i)}$$

Implicit-Target Loss Regularization (ITL)

- A time-varying regularization penalty for improving generalization performance.

$$\tilde{L}(\theta) = L(\theta) + R(\theta)$$
$$= L(\theta) + \lambda \sum_{i=1}^{K} \alpha_t^{(i)} \| I_t - m_t^{(i)} \|^2$$

- Loss is augmented by the distance between each memory cell w.r.t. some desired target vector at time t (e.g. from a teacher network).
  - loss from each memory cell is weighted by the attention value.

Relationship to L2 Regularization

- Scaled version of L2 regularization when $I_t = \vec{0}$ and attention is evenly distributed.

$$
\begin{aligned}
R(\theta) &= \lambda \sum_{i=1}^{K} \alpha_t^{(i)} \| I_t - m_t^{(i)} \|^2 \\
&= \lambda \sum_{i=1}^{K} \frac{1}{K} \| \vec{0} - m_t^{(i)} \|^2 \\
&= \tilde{\lambda} \sum_{i=1}^{K} \| m_t^{(i)} \|^2
\end{aligned}
$$

Choosing the Implicit-Target

- Can use the mean of the memory cell vectors to avoid additional supervision:

$$\tilde{L}(\theta) = L(\theta) + \lambda \sum_{i=1}^{K} \alpha_t^{(i)} \| o_t - m_t^{(i)} \|^2$$

- Regularization term disappears when:
  - a single memory cell is activated.
  - memory cells are identical.
- Interesting results when used with dropout applied to the memory cells…
- **Pulls all memory cells closer to the weighted-mean memory vector.**

Experimental Results

Evaluation with Perplexity (PPL)

- The **perplexity** of a language model is given by:

$$\text{Perplexity} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log P(w_t|w_{t-1},\ldots,w_1)\right)$$

- Measures how good the model is at generating words from the training corpus.
  - **Lower is better.**

Penn Tree Bank: Experimental Setup

- Penn Tree Bank (PTB) corpus was used for language modelling experiments.
  - Roughly 1 million words.
  - consists mainly of text related to finance, politics and business.
  - many existing baseline results from past research studies.
- AMN model contained a single controller and five memory cells.
  - recurrent layer for each memory cell used 100 (or 500) GRUs.
- RNN/GRU/LSTM baselines had a similar number of hidden units and model parameters.

PPL on Penn Tree Bank Corpus

Model	Train	Valid
AMN	82	144
AMN + Implicit	77	141
AMN + Drop-Mem + Implicit	67	104
AMN + Anneal + Drop-Mem + Implicit	70	**103**

Table 6.3 Perplexity for AMN trained with ITL. Lower is better.

PPL on Penn Tree Bank Corpus

Model	Num. Param	Train	Valid	Eval
Large-RNN + Dropout	16m	81	146	139
Large-GRU + Dropout	18m	67	115	114
Large-LSTM + Dropout	20m	65	127	117
Large-AMN + Anneal + Drop-Mem	19m	64	102	96
Large-AMN + Anneal + Drop-Mem + ITL	19m	55	98	**91**

Table 6.4 Perplexity results for large models.

Comparison with Existing Work

Model	Hidd.	Valid	Eval
KN-5 (Mikolov et. al 2012)	NA	148	141
RNN + LDA (Mikolov et. al 2012)	100	132	126
TopicRNN (Dieng et. al 2017)	100	129	122
TopicGRU (Dieng et. al 2017)	100	118	112
TopicLSTM (Dieng et. al 2017)	100	126	118
AMN + Anneal + Drop-Mem + Implicit	100	103	**95**

Table 1: Comparison with explicit neural topic-models on PTB.

BBC Multi-Genre Broadcast News: Experimental Setup

- BBC Multi-Genre Broadcast News (MGB) corpus consists of subtitles drawn from BBC broadcasts.
  - 12 million words.
  - highly-structured data with many genres/topics dependencies.
  - uses conversational-style English.
- Same experimental setup as PTB.

PPL on MGB Corpus

Model	Num. Param	Train	Valid
RNN	43m	98	113
GRU	44m	62	83
AMN	44m	70	**73**

Table 2: Perplexity for 12M MGB.

Implicit Topic Modelling on MGB

- Rank words based on the mean attention value for the j^th memory cell.
  - Tells us which words led to high activation for a given memory cell.

Genre	Vocab Size	Mem1	Mem2	Mem3	Mem4	Mem5
news	45247	0.47	0.48	0.53	0.48	0.51
events	24923	0.25	0.31	0.34	0.30	0.29
competition	32677	0.42	0.28	0.45	0.41	0.34
childrens	26703	0.38	0.33	0.24	0.22	0.22
advice	29959	0.40	0.33	0.31	0.35	0.34
documentary	36262	0.48	0.49	0.42	0.51	0.45
comedy	19961	0.18	0.16	0.20	0.17	0.20

Table 6.7 Percentage of top-100 words in memcell word ranking that appears in the vocabulary of each genre-specific corpus. Values indicating high memcell topic-specialization are highlighted.

Implicit Topic Modelling on MGB

Mem1	Mem2	Mem3	Mem4	Mem5
(15) announcer	(1) resurrect	(1) choosy	(4) annihilation	(1) orchestrate
(16) inventiveness	(2) residue	(2) formalities	(5) slashing	(6) socialising
(24) storytellers	(3) marauder	(8) dreamcoat	(17) grimness	(28) romanticised
(27) wowed	(5) naturalistic	(15) dumbstruck	(20) feuds	(29) candlelit
(29) bamboozled	(13) deathbed	(21) emmy	(28) panicky	(37) uninterrupted
(42) phrasing	(21) undertakers	(41) resplendent	(43) legionnaires	(39) amour

Fig. 6.25 Examples of high-rank words in each memcell word ranking (with rank number).

Evaluation with Word Error Rate (WER)

- Word error rate is given by:

$$\text{WER} = \frac{\text{Sub} + \text{Del} + \text{Ins}}{\text{Number of Words}}$$

- Measures the number of substitution, deletion and insertion errors made (according to the Levenshtein/edit distance).
  - **Lower is better.**
- PPL and WER tend to be positively correlated.

WER on Multi-Genre Broadcast News Corpus

- WER obtained by n-best rescoring on MGB for speech recognition.
  - Sentence hypotheses were re-ranked according to language model probability.

Model	WER
3-gram	28.5
RNN + 3-gram	27.8
GRU + 3-gram	27.2
AMN + 3-gram	27.0

Table 7.3 WER from 100-best rescoring on dev17a.

Conclusions and Future Work

Conclusions

- AMN is an effective model for language modeling.
  - Behavior of model is interpretable via attention-mechanism and generally makes sense.
  - Using ITL improves generalization performance.
- Obtains PPL gains on PTB and both PPL/WER gains on MGB over competitive baselines.
- Can learn an attention mechanism that performs implicit topic modelling.

Future Work

- Incorporate a global memory module for remembering information with very long time-dependencies.
  - e.g. word-inputs from over 100 time-steps ago.
- Apply ITL regularization towards improving RNN performance.
  - can be used to enforce smooth-trajectory constraints on the hidden state.

Questions?